Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses

Anupam Das, Nikita Borisov and Matthew Caesar
University of Illinois at Urbana-Champaign
{das17, nikita, caesar}@illinois.edu

***Abstract*—Modern smartphones contain motion sensors, such as accelerometers and gyroscopes. These sensors have many useful applications; however, they can also be used to uniquely identify a phone by measuring anomalies in the signals, which are a result of manufacturing imperfections. Such measurements can be conducted surreptitiously by web page publishers or advertisers and can thus be used to track users across applications, websites, and visits.**

We analyze how well sensor fingerprinting works under real-world constraints. We first develop a highly accurate fingerprinting mechanism that combines multiple motion sensors and makes use of inaudible audio stimulation to improve detection. We evaluate this mechanism using measurements from a large collection of smartphones, in both lab and public conditions. We then analyze techniques to mitigate sensor fingerprinting either by calibrating the sensors to eliminate the signal anomalies, or by adding noise that obfuscates the anomalies. We evaluate the impact of calibration and obfuscation techniques on the classifier accuracy; we also look at how such mitigation techniques impact the utility of the motion sensors.

I. Introduction

Smartphones are equipped with motion sensors, such as accelerometers and gyroscopes, that are available to applications and websites, and enable a variety of novel uses. These same sensors, however, can threaten user privacy by enabling *sensor fingerprinting*. Manufacturing imperfections result in each sensor having unique characteristics in their produced signal. These characteristics can be captured in the form of a fingerprint and be used to track users across repeated visits. The sensor fingerprint can be used to supplement other privacy-invasive tracking technologies, such as cookies, or canvas fingerprinting [1]. Since the fingerprint relies on the physical characteristics of a particular device, it is immune to defenses such as clearing cookies and private browsing modes.

We carry out a detailed investigation into the feasibility of fingerprinting motion sensors in smartphones. Practical fingerprinting faces several challenges. During a typical web browsing session, a smartphone is either held in a user's hand, resulting in noisy motion inputs, or is resting on a flat surface, minimizing the amount of sensor input. Additionally, web APIs for accessing motion sensor data have significantly lower resolution than is available to the operating systems and applications. We show that, using machine learning techniques, it is possible to combine a large number of features from both the accelerometer and gyroscope sensor streams and produce highly accurate classification despite these challenges. In some cases, we can improve the classifier accuracy by using an inaudible sound, played through the speakers, to stimulate the motion sensors. We evaluate our techniques in a variety of lab settings; additionally, we collected data from volunteer participants over the web, capturing a wide variety of smartphone models and operating systems. In our experiments, a web browsing session lasting in the orders of 30–40 seconds is sufficient to generate a fingerprint that can be used to recognize the phone in the future with only 5–8 seconds worth of web browsing session.

We next investigate two potential countermeasures to sensor fingerprinting. First, we consider the use of *calibration* to eliminate some of the errors that result from manufacturing imperfections. Promisingly, we find that calibrating the accelerometer is easy and has a significant impact on classification accuracy. Gyroscope calibration, however, is more challenging without specialized equipment, and attempts to calibrate the gyroscope by hand do not result in an effective countermeasure.

An alternative countermeasure is *obfuscation*, which introduces additional noise to the sensor readings in the hopes of hiding the natural errors. Obfuscation has the advantage of not requiring a calibration step; we find that by adding noise that is similar in magnitude to the natural errors that result from manufacturing imperfection, we can reduce the accuracy of fingerprinting more effectively than by calibration. We also investigate the possibility of using higher magnitude noise, as well as adding temporal disturbances to obfuscate frequency domain features. At high levels of noise, fingerprinting accuracy is greatly reduced, though such noise is likely to impair the utility of motion sensors.

Roadmap. The remainder of this paper is organized as follows. We present background information and related work in Section II. In Section III, we briefly discuss why accelerometers and gyroscopes can be used to generate unique fingerprints. In Section IV, we describe the different temporal and spectral features considered in our experiments, along with the classification algorithms and metrics used in our evaluations. We present our fingerprinting results in Section V. Section VI describes our countermeasure techniques to sensor fingerprinting. Section VII discusses some limitations of our approach. Finally, we conclude in Section VIII.

NDSS '16, 21-24 February 2016, San Diego, CA, USA

http://dx.doi.org/10.14722/ndss.2016.23390

II. Fingerprinting Background

Human fingerprints, due to their unique nature, are a very popular tool used to identify people in forensic and biometric applications [4], [5]. Researchers have long sought to find an equivalent of fingerprints in computer systems by finding characteristics that can help identify an individual device. Such fingerprints exploit variations in both the hardware and software of devices to aid in identification.

As early as 1960, the US government used unique transmission characteristics to track mobile transmitters [6]. Later, with the introduction of cellular network researchers were able to successfully distinguish transmitters by analyzing the spectral characteristics of the transmitted radio signal [7]. Researchers have suggested using radio-frequency fingerprints to enhance wireless authentication [8], [9], as well as localization [10]. Others have leveraged the minute manufacturing imperfections in network interface cards (NICs) by analyzing the radio-frequency of the emitted signals [11], [12]. Computer clocks have also been used for fingerprinting: Moon et al. showed that network devices tend to have a unique and constant clock skews [13]; Kohno et al. exploited this to distinguish network devices through TCP and ICMP timestamps [14].

Software can also serve as a distinguishing feature, as different devices have a different installed software base. Researchers have long been exploiting the difference in the protocol stack installed on IEEE 802.11 compliant devices. Desmond et al. [15] have looked at distinguishing unique devices over Wireless Local Area Networks (WLANs) simply by performing timing analysis on the 802.11 probe request packets. Others have investigated subtle differences in the firmware and device drivers running on IEEE 802.11 compliant devices [16]. 802.11 MAC headers have also been used to uniquely track devices [17]. Moreover, there are well-known open source toolkits like Nmap [18] and Xprobe [19] that can remotely fingerprint an operating system by analyzing unique responses from the TCP/IP networking stack.

a) Browser Fingerprinting: A common application of fingerprinting is to track a user across multiple visits to a website, or a collection of sites. Traditionally, this was done with the aid of cookies explicitly stored by the browser. However, privacy concerns have prompted web browsers to implement features that clear the cookie store, as well as private browsing modes that do not store cookies long-term. This has prompted site operators to develop other means of uniquely identifying and tracking users. Eckersley's Panopticon project showed that many browsers can be uniquely identified by enumerating installed fonts and other browser characteristics, easily accessible via JavaScript [20]. A more advanced technique uses HTML5 canvas elements to fingerprint the fonts and rendering engines used by the browser [1]. Others have proposed the use of performance benchmarks for differentiating between JavaScript engines [21]. Lastly, browsing history can to used to profile and track online users [22]. Numerous studies have found evidence of these and other techniques being used in the wild [23]–[25]. A number of countermeasures to these techniques exist; typically they disable or restrict the ability of a website to probe the characteristics of a webbrowser. Nikiforakis et al. propose using random noise to make fingerprints non-deterministic which essentially breaks linkability across multiple visits [26]. We expect that smartphones are less susceptible to browser fingerprinting due to a more integrated hardware and software base resulting in less variability, though we are unaware of an exploration of smartphone browser fingerprinting.

Alternative to cookies people have also looked at leveraging device IDs such as Unique Device Identifier (UDID) for Apple products and International Mobile Station Equipment Identity (IMEI) for general mobile phones, to track devices across multiple visits. However, these device IDs are not always accessible (Apple ceased the use of UDID since iOS 6 [27]) and even if it is accessible, in most cases it requires explicit permission to access such device ID (on Android accessing IMEI requires a special permission [28]).

b) Sensor Fingerprinting: Smartphones do, however, possess an array of sensors that can be used to fingerprint them. Two studies have looked at fingerprinting smartphone microphones and speakers [29], [30]. These techniques, however, require access to the microphone, which is typically controlled with a separate permission due to the obvious privacy concerns with the ability to capture audio. Bojinov et al. [3] consider using accelerometers, which are not considered sensitive and do not require a separate permission. Their techniques, however, rely on having the user perform a calibration of the accelerometer (see Section VI-A), the parameters of which are used to distinguish phones. Dey et al. [2] apply machine learning techniques to create an accelerometer fingerprint; most of their analysis focuses on using the vibration motor to stimulate the accelerometer, but they perform an experiment with 25 stationary phones and on average they achieve approximately 88% precision and recall.

In contrast, our work studies phones that are in a natural web-browsing setting, either in a user's hand or resting on a flat surface. Additionally, we consider the simultaneous

TABLE I: Comparison with other works

Work	Sensors[a]	Settings	Stimulation	Features Explored	Features Used	# of Devices	Results (≈)
[2]	A	Lab	Vibration	80	36	107[b]	99% Accuracy
[2]	A	Lab	None	80	36	25	88% F-score
[3]	A	Lab	Flip phone	2	2	33	100% Accuracy
[3]	A	Public	Flip phone	2	2	3583[c]	15.1% Accuracy
Our Work	A,G	Lab	None	100	70	30	99% F-score
Our Work	A,G	Public	None	100	70	63	95% F-score
Our Work	A,G	Lab+Public	None	100	70	93	96% F-score
Our Work	A,G	Lab	Phone in hand	100	70	30	93% F-score
Our Work	A,G	Lab	Phone in hand+Audio	100	70	30	98% F-score

[a] here **'A'** means accelerometer and **'G'** refers to gyroscope
[b] 80 external chips, 25 phones and 2 tablets
[c] considering only devices with two submissions

use of both accelerometer and gyroscope to produce a more *accurate fingerprint*. Inspired by prior work that uses the gyroscope to recover audio signals [31], we also stimulate the gyroscope with an inaudible tone. Finally, we propose and evaluate several ***countermeasures*** to reduce fingerprinting accuracy without entirely blocking access to the motion sensors. Table I highlights some comparisons with related works. Recently, Song et al. [32] have proposed reducing accelerometer accuracy as a means of defense against tap inference on smartphones. Their approach involves hiding small changes in accelerometer reading by reporting a constant accelerometer value of $1g$. We propose one similar technique where we calibrate motion sensors so that they report similar constant readings. However, as we will later on show that such an approach is not sufficient to hide uniqueness among gyroscope sensors. We, therefore, explore several obfuscation techniques in this paper.

III. A Closer Look at Motion Sensors

In this section we briefly take a closer look at motion sensors like accelerometer and gyroscope that are embedded in today's smartphones. This will provide an understanding of how they can be used to uniquely fingerprint smartphones. Accelerometer and gyroscope sensors in modern smartphones are based on Micro Electro Mechanical Systems (MEMS). STMicroelectronics [33] and InvenSense [34] are among the top vendors supplying MEMS-based accelerometer and gyroscope sensor to different smartphone manufacturers [35]. Traditionally, Apple [36], [37][1] and Samsung [39], [40] favor using STMicroelectronics motion sensors, while Google [41], [42] tends to use InvenSense sensors.

A. Accelerometer

Accelerometer is a device that measures proper acceleration. Proper acceleration is different from coordinate acceleration (linear acceleration) as it measures the *g-force*. For example, an accelerometer at rest on a surface will measure an acceleration of $g = 9.81ms^{-2}$ straight upwards, while for a free falling object it will measure an acceleration of zero. MEMS-based accelerometers are based on differential capacitors [43]. Figure 1 shows the internal architecture of a MEMS-based accelerometer. As we can see there are several pairs of fixed electrodes and a movable seismic mass. Under zero force the distances d_1 and d_2 are equal and as a result the two capacitors are equal, but a change in force will cause the movable seismic mass to shift closer to one of the fixed electrodes (i.e., $d_1 \neq d_2$) causing a change in the generated capacitance. This difference in capacitance is detected and amplified to produce a voltage proportional to the acceleration. The slightest gap difference between the structural electrodes, introduced during the manufacturing process, can cause a change in the generated capacitance. Also the flexibility of the seismic mass can be slightly different from one chip to another. These form of minute imprecisions in the electro-mechanical structure induce subtle imperfections in accelerometer chips.

B. Gyroscope

Gyroscope measures the rate of rotation (in $rads^{-1}$) along the device's three axes. MEMS-based gyroscopes use the

[1]iPhone 6 has been reported to use sensors made by InvenSense [38]

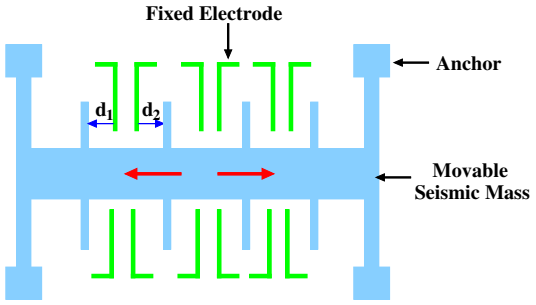

Fig. 1: Internal architecture of a MEMS accelerometer. Differential capacitance is proportional to the applied acceleration.

Coriolis effect to measure the angular rate. Whenever an angular velocity of $\hat{\omega}$ is exerted on a moving mass of weight m, and velocity $\hat{v}$, the object experiences a Coriolis force in a direction perpendicular to the rotation axis and to the velocity of the moving object (as shown in figure 2). The Coriolis force is calculated by the following equation $\hat{F} = -2m\hat{\omega} \times \hat{v}$. Generally, the angular rate ($\hat{\omega}$) is measured by sensing the magnitude of the Coriolis force exerted on a vibrating proof-mass within the gyro [44]–[46]. The Coriolis force is sensed by a capacitive sensing structure where a change in the vibration of the proof-mass causes a change in capacitance which is then converted into a voltage signal by the internal circuitry. Again the slightest imperfection in the electro-mechanical structure will introduce idiosyncrasies across chips.

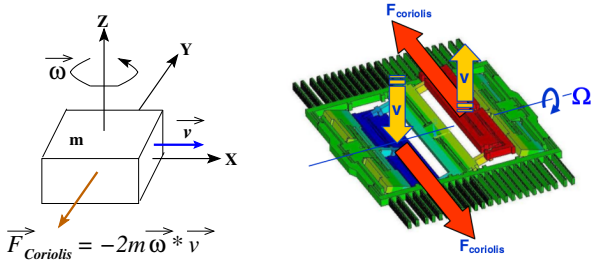

Fig. 2: MEMS-based gyros use *Coriolis force* to compute angular velocity. The Coriolis force induces change in capacitance which is proportional to the angular velocity.

IV. Features and Classification Algorithms

Here, we describe the data preprocessing step and the features used in generating the sensor fingerprint. We also discuss the classification algorithms and metrics used in our evaluation.

A. Data Preprocessing

Data from motion sensors can be thought of as a stream of timestamped real values. For both accelerometer and gyroscope we obtain values along three axes. So, for a given timestamp, t, we have two vectors of the following form: $\vec{a}(t) = (a_x, a_y, a_z)$ and $\vec{\omega}(t) = (\omega_x, \omega_y, \omega_z)$. The accelerometer values include gravity, i.e., when the device is stationary lying flat on top of a surface we get a value of $9.81ms^{-2}$ along the z-axis. We convert the acceleration vector into a scalar by taking its magnitude: $|\vec{a}(t)| = \sqrt{a_x^2 + a_y^2 + a_z^2}$. This technique

TABLE II: Explored temporal and spectral features

#	Domain	Feature	Description
1	Time	Mean	The arithmetic mean of the signal strength at different timestamps
2		Standard Deviation	Standard deviation of the signal strength
3		Average Deviation	Average deviation from mean
4		Skewness	Measure of asymmetry about mean
5		Kurtosis	Measure of the flatness or spikiness of a distribution
6		RMS	Square root of the arithmetic mean of the squares of the signal strength at various timestamps
7		Max	Maximum signal strength
8		Min	Minimum signal strength
9		ZCR	The rate at which the signal changes sign from positive to negative or back
10		Non-Negative count	Number of non-negative values
11	Frequency	Spectral Centroid	Represents the center of mass of a spectral power distribution
12		Spectral Spread	Defines the dispersion of the spectrum around its centroid
13		Spectral Skewness	Represents the coefficient of skewness of a spectrum
14		Spectral Kurtosis	Measure of the flatness or spikiness of a distribution relative to a normal distribution
15		Spectral Entropy	Captures the peaks of a spectrum and their locations
16		Spectral Flatness	Measures how energy is spread across the spectrum
17		Spectral Brightness	Amount of spectral energy corresponding to frequencies higher than a given cut-off threshold
18		Spectral Rolloff	Defines the frequency below which 85% of the distribution magnitude is concentrated
19		Spectral Roughness	Average of all the dissonance between all possible pairs of peaks in a spectrum
20		Spectral Irregularity	Measures the degree of variation of the successive peaks of a spectrum
21		Spectral RMS	Square root of the arithmetic mean of the squares of the signal strength at various frequencies
22		Low-Energy-Rate	The percentage of frames with RMS power less than the average RMS power for the whole signal
23		Spectral flux	Measure of how quickly the power spectrum of a signal changes
24		Spectral Attack Time	Average rise time to spectral peaks
25		Spectral Attack Slope	Average slope to spectral peaks

discards some information, but has the advantage of making the accelerometer data independent of device orientation; e.g., if the device is stationary the acceleration magnitude will always be around $9.81ms^{-2}$, whereas the reading on each individual axis will vary greatly (by +/- $1g$) depending on how the device is held. For the gyroscope we consider data from each axis as a separate stream, since there is no corresponding baseline rotational acceleration. In other words, if the device is stationary the rotation rate across all three axes should be close to 0 $\texttt{rad}s^{-1}$, irrespective of the orientation of the device. Thus, our model considers four streams of sensor data in the form of $\{|\vec{a}(t)|, \omega_x(t), \omega_y(t), \omega_z(t)\}$.

For all data streams, we also look at frequency domain characteristics. But since the browser, running as one of many applications inside the phone, makes API calls to collect sensor data the OS might not necessarily respond in a synchronized manner[2]. This results in non-equally spaced data points. We, therefore, use cubic-spline interpolation [47] to construct new data points such that $\{|\vec{a}(t)|, \omega_x(t), \omega_y(t), \omega_z(t)\}$ become equally-spaced.

B. Temporal and Spectral Features

To summarize the characteristics of a sensor data stream, we explore a total of 25 features consisting of 10 temporal and 15 spectral features (listed in Table II). All of these features have been well documented by researchers in the past. A detailed description of each feature is available in our technical report [48].

C. Classification Algorithms and Metrics

Classification Algorithms: Once we have features extracted from the sensor data, we use supervised learning to identify the source sensor. Any supervised learning classifier has two main phases: training phase and testing phase. During training, features from all smartphones (i.e., labeled data) are used to train the classifier. In the test phase, the classifier predicts the most probable class for a given (unseen) feature vector. We evaluate the performance of the following classifiers — Support Vector Machine (SVM), Naive-Bayes classifier, Multiclass Decision Tree, k-Nearest Neighbor (k-NN), Quadratic Discriminant Analysis classifier and Bagged Decision Trees (Matlab's Treebagger model) [49]. We found that in general ensemble based approaches like Bagged Decision Trees outperform the other classifiers. We report the maximum achievable accuracies from these classifiers in the evaluation Section V.

Evaluation metrics: For evaluation metric we use standard multi-class classification metrics like—*precision*, *recall*, and *F-score* [50]—in our evaluation. Assuming there are n classes, we first compute the true positive (TP) rate for each class, i.e., the number of traces from the class that are classified correctly. Similarly, we compute the false positive (FP) and false negative (FN) as the number of wrongly accepted and wrongly rejected traces, respectively, for each class i ($1 \leq i \leq n$). We then compute precision, recall, and the F-score for each class using the following equations:

$$\text{Precision, } Pr_i = TP_i/(TP_i + FP_i) \quad (1)$$

$$\text{Recall, } Re_i = TP_i/(TP_i + FN_i) \quad (2)$$

$$\text{F-Score, } F_i = (2 \times Pr_i \times Re_i)/(Pr_i + Re_i) \quad (3)$$

The F-score is the harmonic mean of precision and recall; it provides a good measure of overall classification performance, since precision and recall represent a trade-off: a more conservative classifier that rejects more instances will have higher precision but lower recall, and vice-versa. To obtain the overall performance of the system we compute average values in the

[2]Depending on the load and other applications running, OS might prioritize such API calls differently.

following way:

$$\text{Avg. Precision, } AvgPr = \frac{\sum_{i=1}^{n} Pr_i}{n} \quad (4)$$

$$\text{Avg. Recall, } AvgRe = \frac{\sum_{i=1}^{n} Re_i}{n} \quad (5)$$

$$\text{Avg. F-Score, } AvgF = \frac{2 \times AvgPr \times AvgRe}{AvgPr + AvgRe} \quad (6)$$

V. Fingerprinting Evaluation

In this section we first describe our experimental setup (Section V-A). We then explore features to determine the minimal subset of features required to obtain high classification accuracy (Section V-B). Lastly, we evaluate our fingerprinting approach under a controlled lab setting (Section V-C), an uncontrolled real-world setting (Section V-D) and a combination of both settings (Section V-E).

A. Experimental Setup

Given that mobile accounts for a third of all global web pages served [51], our experimental setup consists of developing our own web page to collect sensor data[3]. We use a simple Javascript (code snippet available in Appendix A) to access accelerometer and gyroscope data. However, since we collect data through the browser the maximum obtainable sampling frequency is lower than the available hardware sampling frequency (restricted by the underlying OS). Table III summarizes the sampling frequencies obtained from the top 5 mobile browsers [52][4]. We use a Samsung Galaxy S3 and iPhone 5 to test the sampling frequency of the different browsers. Table III also highlights the motion sensors that are accessible from the different browsers. We see that Chrome provides the best sampling frequency while the default Android browser is the most restrictive browser in terms of not only sampling frequency but also access to different motion sensors. However, Chrome being the most popular mobile browser [53], we collect data using the Chrome browser.

TABLE III: Sampling frequency from different browsers

OS	Browser	Sampling Frequency (∼Hz)	Accessible Sensors[a]
Android 4.4	Chrome	100	A,G
	Android	20	A
	Opera	40	A,G
	UC Browser	20	A,G
	Standalone App [54]	200	A,G
iOS 8.1.3	Safari	40	A,G
	Standalone App [55]	100	A,G

[a]here **'A'** means accelerometer and **'G'** refers to gyroscope

We start off our data collection from 30 lab-smartphones. Table IV lists the distribution of the different smartphones from which we collect sensor data. Now, as gyroscopes react to audio stimulation we collect data under three different background audio settings: *no audio*, *an inaudible 20 kHz sine wave*, or *a popular song*. In the latter two scenarios, the corresponding audio file plays in the background of the browser while data is being collected. Under each setting we collect 10 samples where each sample is about 5 to 8 seconds worth of data. Now, since our fingerprinting approach aims to capture the inherent imperfections of motion sensors, we need to keep the sensors stationary while collecting data. Therefore, by default, we have the phone placed flat on a surface while data is being collected, unless explicitly stated otherwise. We, however, do test our approach for the scenario where the user is holding the smartphone in his/her hand while sitting down.

For training and testing the classifiers we *randomly* split the dataset in such a way that 50% of data from each device goes to the training set while the remaining 50% goes to the test set. To prevent any bias in the selection of the training and testing set, we randomize the training and testing set 10 times and report the average F-score. We also compute the 95% confidence interval, but we found it to be less than 1% in most cases and hence do not report them in such cases. For analyzing and matching fingerprints we use a desktop machine with an Intel i7-2600 3.4GHz processor with 12GiB RAM. We found that the average time required to match a new fingerprint was around 10–100 ms.

TABLE IV: Types of phones used

Maker	Model	Quantity
Apple	iPhone 5	4
	iPhone 5s	3
Samsung	Nexus S	14
	Galaxy S3	4
	Galaxy S4	5
Total		30

B. Feature Exploration and Selection

At first glance, it might seem that using all features to identify the device is the optimal strategy. However, including too many features can worsen performance in practice, due to their varying accuracies and potentially-conflicting signatures. We, therefore, explore all the features and determine the subset of features that optimize our fingerprinting accuracy. For temporal features, no transformation of the data stream is required, but for spectral features we first convert the non-equally spaced data stream into a fixed-spaced data stream using cubic spline interpolation. We interpolate at a sampling rate of 8kHz[5]. Then, we use the following signal analytic tools and modules: *MIRtoolbox* [56] and *Libxtract* [57] to extract spectral features. We next look at feature selection where we explore different combinations of features to maximize our fingerprinting accuracy. We use the FEAST toolbox [58] and utilize the *Joint Mutual Information* criterion (JMI criterion is known to provide the best tradeoff in terms of accuracy, stability, and flexibility with small data samples [59]) for ranking the features.

Figure 3 shows the results of our feature exploration for the 30 lab-smartphones. We see that when using only accelerometer data the F-score seems to flatten after considering the top 10 features. For gyroscope data we see that using all 75 features (25 per data stream) achieves the best result. And finally when we combine both accelerometer and gyroscope features, the

[3]http://datarepo.cs.illinois.edu/DataCollectionHowPlaced.html

[4]Computed the avg. time to obtain 100 samples. http://datarepo.cs.illinois.edu/SamplingFreq.html

[5]Although up-sampling the signal from ∼100 Hz to 8 kHz does not increase the accuracy of the signal, it does make direct application of standard signal processing tools more convenient.

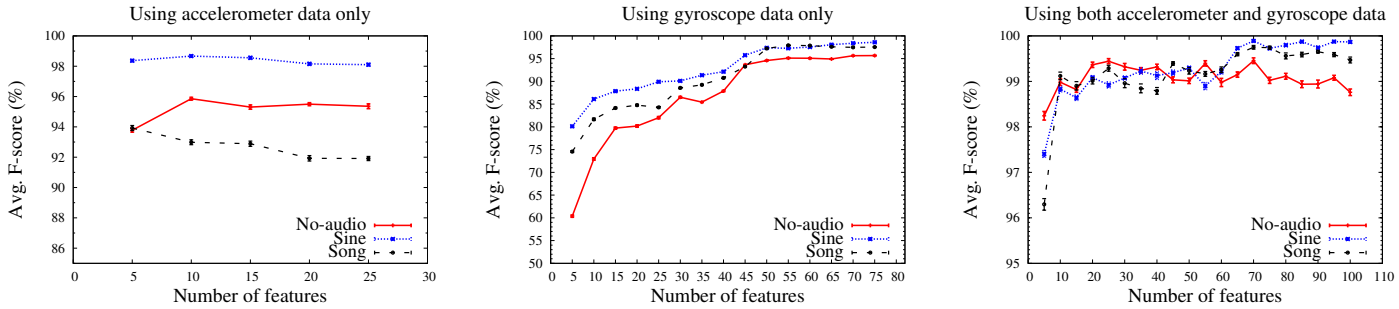

Fig. 3: Exploring the number optimal features for different sensors. a) For accelerometer using more than top 10 features leads to diminished returns, b) For gyroscope all 75 features contribute to obtaining improved accuracy, c) For the combined sensor data using more than 70 features leads to diminished returns.

top 70 features (from a total of 100 features) seems to provide the best fingerprinting accuracy. Among these top 70 features we found that 21 of them came from accelerometer features and the remaining 49 came from gyroscope features. In terms of the distribution between temporal and spectral features, we found that spectral features dominated with 44 of the top 70 features being spectral features. We use these subset of features in all our later evaluations.

C. Results From Lab Setting

First, we look at fingerprinting smartphones under lab setting to demonstrate the basic viability of the attack. For this purpose we keep smartphones stationary on top of a flat surface. Table V summarizes our results. We see that we can almost correctly identify all 30 smartphones for all three scenarios by combining the accelerometer and gyroscope features. Even when devices are kept in the hand of the user we can successfully identify devices with an F-score of greater than 93%. While the benefit of the background audio stimulation is not clear from the table, we will later on show that audio stimulation do in fact enhance fingerprinting accuracy in the presence of countermeasure techniques like sensor calibration and data obfuscation (more in Section VI). Overall these results indicate that it is indeed possible to fingerprint smartphones through motion sensors.

TABLE V: Average F-score under lab setting

Device Placed	Stimulation	Avg. F-score (%)		
		Accelerometer	Gyroscope	Accelerometer+Gyroscope
On Desk	No-audio	96	95	99
	Sine	98	99	100
	Song	93	98	100
In Hand	No-audio	88	83	93
	Sine	88	94	98
	Song	84	89	95

D. Results From Public Setting

After gaining promising results from our relatively small-scale lab setting, we set out to expand our data collection process to real-world public setting. We invited people to voluntarily participate in our study by visiting our web page[6] and following a few simple steps to provide us with sensor

[6]Screenshots of the data collection page is available in Appendix B. We obtained approval from our Institutional Research Board (IRB) to perform the data collection.

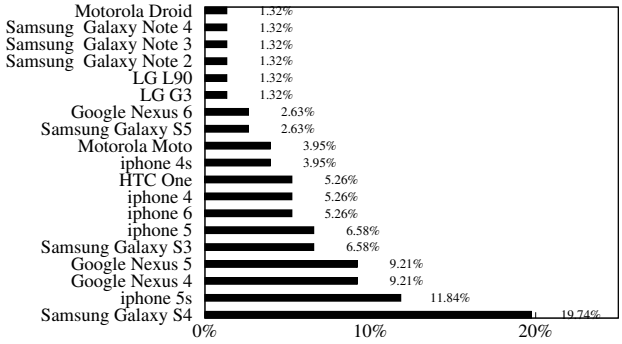

Fig. 4: Distribution of participant device model.

data. We recruited participants through email and online social networks. We asked participants to provide data under two settings: no-audio setting and the inaudible sine-wave setting (we avoid the background song to make the experience less bothersome for the user). Each setting collected sensor data for about one minute, requiring a total of two minutes of participation. On average, we had around 10 samples per setting per device. Our data-gathering web page plants a cookie in the form of a large random number (acting as a unique ID) in the user's browser, which makes it possible to correlate data points coming from the same device. Over the course of two weeks, we received data from a total of 76 devices. However, some participants did not follow all the steps and as a result we were able to use only 63 of the 76 submissions. Figure 4 shows the distribution of the different devices that participated in our study.

Next, we apply our fingerprinting approach on the public data set. Table VI shows our findings. Compared to the results from our lab setting, we see a slight decrease in F-score but even then we were able to obtain an F-score of 95%. Again, the benefit of the audio stimulation is not evident from these results, however, their benefits will become more visible in the later sections when we discuss countermeasure techniques.

E. Results From Combined Setting

Finally, we combine our lab data with the publicly collected data to give us a combined dataset containing 93 different smartphones. We apply the same set of evaluations on this combined dataset. Table VII highlights our findings. Again,

TABLE VI: Average F-score under public setting where smartphones were kept *on top of a desk*

Stimulation	Avg. F-score (%)		
	Accelerometer	Gyroscope	Accelerometer+Gyroscope
No-audio	86	87	95
Sine	85	87	92

we see that combining features from both sensors provides the best result. In this case we obtained an F-score of 96%. All these results suggest that smartphones can be successfully fingerprinted through motion sensors.

TABLE VII: Average F-score under both lab and public setting where smartphones were kept *on top of a desk*

Stimulation	Avg. F-score (%)		
	Accelerometer	Gyroscope	Accelerometer+Gyroscope
No-audio	85	89	96
Sine	89	89	95

F. Sensitivity Analysis

1) Varying the Number of Devices: We evaluate the accuracy of our classifier while varying the number of devices. We pick a subset of n devices in our dataset and perform the training and testing steps for this subset. For each value of n, we repeat the experiment 10 times, using a different random subset of n devices each time. In this experiment we only consider the use of both accelerometer and gyroscope features, since those produce the best performance (as evident from our previous results), and focus on the no-audio and sine wave background scenarios. Figure 5 shows that the F-score generally decreases with large number of devices, which is expected as an increased number of labels makes classification more difficult. But even then scaling from 10 devices to 93 devices the F-score decreases by only 4%. Extrapolating from the graph, we expect classification to remain accurate even for significantly larger datasets.

2) Varying Training Set Size: We also consider how varying the training set size impacts the fingerprinting accuracy. For this experiment we vary the ratio of training and testing set size. For this experiment we only look at data from our lab setting as some of the devices from our public setting did not have exactly 10 samples. We also consider the setting where there is no background audio stimulation and use the combined features of accelerometer and gyroscope. Figure 6 shows our findings. While an increased training size improves classification accuracy, even with mere two training samples (of 5–8 seconds each) we can achieve an F-score of 98%, with increased training set sizes producing an F-score of over 99%.

3) Varying Temperature: Here we analyze how temperature impacts the fingerprint of smartphone sensors. For this purpose we collect sensor data under different temperatures. We took one set of readings outside our office building on September 03, 2015 (with temperatures in the range of $91°F$ to $93°F$) while we took another set of readings on October 9, 2015 (with temperatures in the range of $61°F$ to $63°F$). In both cases we also took readings inside the office where temperature was set to around $74°F$ on the thermostat. As these set of experiments were conducted at a later time compared to our other experiments, we were only able to collect data from

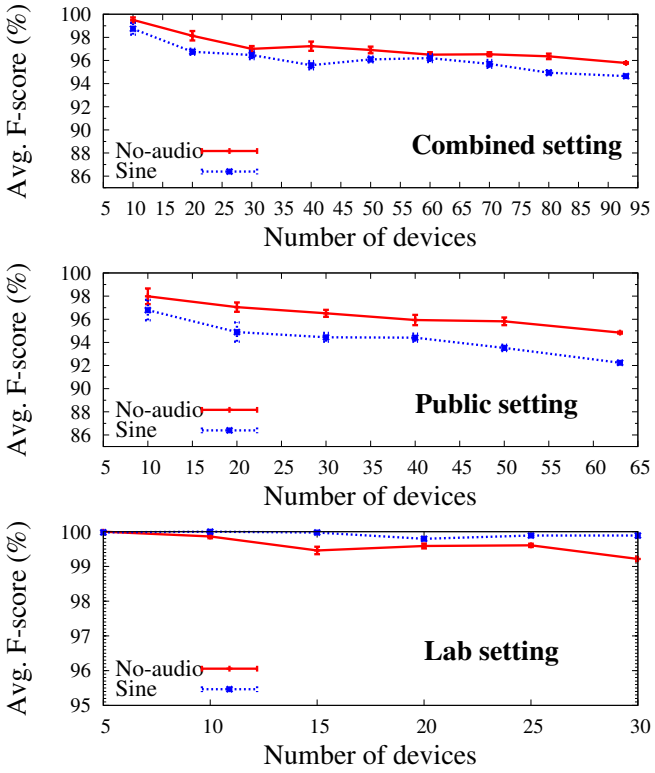

Fig. 5: Average F-score for different numbers of smartphones. F-score generally tends to decrease slightly as more devices are considered.

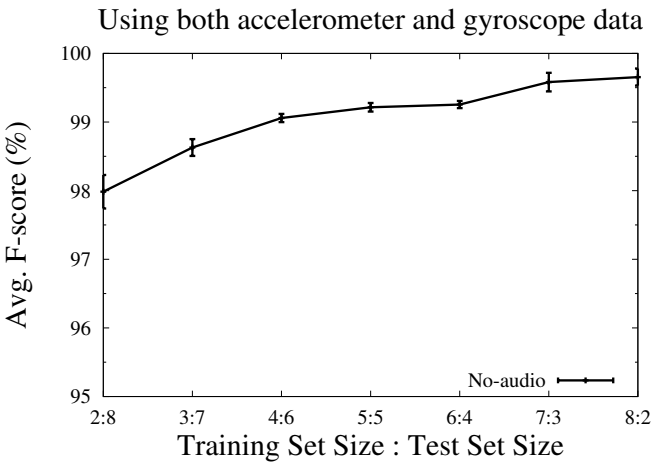

Fig. 6: Average F-score for different ratio of training and testing data. With only two training data we achieved an F-score of 98%.

17 smartphones (as described in Table VIII)[7]. Therefore, the following results for this section are described in the context of only the smartphones specified in Table VIII.

Table IX summarizes our findings. We refer to September 03, 2015 as a *hot* day and October 09, 2015 as a *cold* day. From Table IX we see that temperatures do lower F-score where warmer temperatures cause more discrepancies in the generated fingerprints compared to colder temperatures (as indicated by the red and blue blocks in the table).

[7]We only had access to these 17 smartphones at that time.

TABLE VIII: Types of phones used for analyzing temperature effect

Maker	Model	Quantity
Apple	iPhone 5	4
	iPhone 5s	3
Samsung	Nexus S	3
	Galaxy S3	2
	Galaxy S4	5
Total		17

TABLE IX: Impact of temperature on sensor fingerprinting

No-audio		Test (Avg. F-score in %)			
		Inside (hot)	Outside (hot)	Inside (cold)	Outside (cold)
Train	Inside (hot)	**100**[a]	89	90	92
	Outside (hot)	90	**100**[a]	81	75
	Inside (cold)	89	77	**100**[a]	97
	Outside (cold)	86	82	99	**100**[a]

Sine wave		Test (Avg. F-score in %)			
		Inside (hot)	Outside (hot)	Inside (cold)	Outside (cold)
Train	Inside (hot)	**100**[a]	80	92	91
	Outside (hot)	83	**99**[a]	82	72
	Inside (cold)	88	72	**100**[a]	90
	Outside(cold)	85	69	92	**100**[a]

[a]50% of the data set was used for training and remaining 50% for testing

4) Temporal Stability: We now take a closer look at how the fingerprints evolve over time. For this purpose we reuse data collected from the previous section (Section V-F3). As we collected data inside our lab in two different dates (one on September 03, 2015 and the other on October 09, 2015) we can analyze how sensor fingerprints change over time and how they impact our F-score. Table X summarizes our findings. We see that over time fingerprints do change to some extent, but even then we can achieve an F-score of approximately 90%.

TABLE X: Fingerprinting sensors at different dates

No-audio		Test (Avg. F-score in %)	
		Sept. 03, 2015	Oct. 09,2015
Train	Sept. 03, 2015	**100**[a]	90
	Oct. 09,2015	89	**100**[a]

Sine wave		Test (Avg. F-score in %)	
		Sept. 03, 2015	Oct. 09,2015
Train	Sept. 03, 2015	**100**[a]	92
	Oct. 09,2015	88	**100**[a]

[a]50% of the data set was used for training and remaining 50% for testing

VI. Countermeasures

So far we have focused on showing how easy it is to fingerprint smartphones through motion sensors. We now shift our focus on providing a systematic approach to defending against such fingerprinting techniques. We propose two approaches: sensor calibration and data obfuscation.

A. Calibration

Bojinov et al. [3] observe that their phones have calibration errors, and use these calibration differences as a mechanism to distinguish between them. In particular, they consider an affine error model: $a^M = g \cdot a + o$, where a is the true acceleration along an axis and a^M is the measured value of the sensor. The two error parameters are the offset o (bias away from 0) and the gain g which magnifies or diminishes the acceleration value. Our classification uses many features, but we find that the *mean* signal value is the most discriminating feature for each of the sensor streams, which is closely related to the offset. We therefore explore whether calibrating the sensors will make them more difficult to fingerprint. We note that calibration has a side effect of improving the accuracy of sensor readings and is therefore of independent value. We perform the calibration only on the sensors in our 30 lab smartphones because we felt that calibration is too time consuming for the volunteers[8]. Moreover, we could better control the quality of the calibration process when carried out in the lab.

First, let us briefly describe the sensor coordinate system where the sensor framework uses a standard 3-axis coordinate system to express data values. For most sensors, the coordinate system is defined relative to the device's screen when the device is held in its default orientation (shown in figure 7). When the device is held in its default orientation, the positive x-axis is horizontal and points to the right, the positive y-axis is vertical and points up, and the positive z-axis points toward the outside of the screen face[9]. We compute offset and gain error in all three axes.

Calibrating the Accelerometer: Considering both offset and gain error, the measured output of the accelerometer ($a^M = [a_x^M, a_y^M, a_z^M]$) can be expressed as:

$$\begin{bmatrix} a_x^M \\ a_y^M \\ a_z^M \end{bmatrix} = \begin{bmatrix} O_x \\ O_y \\ O_z \end{bmatrix} + \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad (7)$$

where $S = [S_x, S_y, S_z]$ and $O = [O_x, O_y, O_z]$ respectively represents the gain and offset errors along all three axes ($a = [a_x, a_y, a_z]$ refers to the actual acceleration). In the ideal world $[S_x, S_y, S_z] = [1, 1, 1]$ and $[O_x, O_y, O_z] = [0, 0, 0]$, but in reality they differ from the desired values. To compute the offset and gain error of an axis, we need data along both the positive and negative direction of that axis (one measures positive $+g$ while the other measures negative $-g$). In other words, six different *static* positions are used where in each position one of the axes is aligned either along or opposite to earth's gravity. This causes the $a = [a_x, a_y, a_z]$ vector to take one of the following six possible values $\{[\pm g, 0, 0], [0, \pm g, 0], [0, 0, \pm g]\}$. For example, if a_{z+}^M and a_{z-}^M are two values of accelerometer reading along the positive and negative z-axis, then we can compute the offset (O_z) and gain (S_z) error using the following equation:

$$S_z = \frac{a_{z+}^M - a_{z-}^M}{2g}, \qquad O_z = \frac{a_{z+}^M + a_{z-}^M}{2} \quad (8)$$

We take 10 measurements along all six directions ($\pm x, \pm y, \pm z$) from all our lab devices as shown in Figure 7. From these measurements we compute the average offset and gain error along all three axes using equation (8). Figure 8 shows a scatter-plot of the errors along $\texttt{z}-\texttt{axis}$ for 30 smartphones (each color code represents a certain make-and-model). We can see that the devices are scattered around all over the plot which signifies that different devices have different amount of offset and gain error. Such unique distinction makes fingerprinting feasible.

[8]Requiring around 12 minutes in total for calibrating both the accelerometer and gyroscope.

[9]Android and iOS consider the positive and negative direction along an axis differently.

Fig. 7: Calibrating accelerometer along three axes. We collect measurements along all 6 directions $(\pm x, \pm y, \pm z)$.

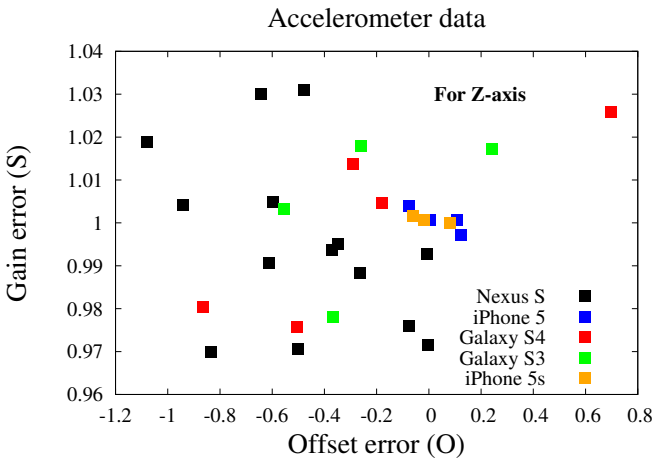

Fig. 8: Accelerometer offset and gain error from 30 smartphones.

Calibrating the Gyroscope: Calibrating gyroscope is a harder problem as we need to induce a fixed angular change to determine the gain error even though the offset error can be computed while keeping the device stationary[10]. Similar to accelerometer we can also represent the measured output of the gyroscope ($\omega^M = [\omega_x^M, \omega_y^M, \omega_z^M]$) using the following equation:

$$\begin{bmatrix} \omega_x^M \\ \omega_y^M \\ \omega_z^M \end{bmatrix} = \begin{bmatrix} O_x \\ O_y \\ O_z \end{bmatrix} + \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (9)$$

where again $S = [S_x, S_y, S_z]$ and $O = [O_x, O_y, O_z]$ respectively represents the gain and offset errors along all three axes. Here, $\omega = [\omega_x, \omega_y, \omega_z]$ represents the ideal/actual angular velocity. Ideally all gain and offset errors should be equal to 1 and 0 respectively. But in the real world when the device is rotated by a fixed amount of angle, the measured angle tends to deviate from the actual angular displacement (shown in figure 9(a)). This impacts any system that uses gyroscope for angular-displacement measurements.

To calibrate gyroscope we again need to collect data along all six different directions $(\pm x, \pm y, \pm z)$ individually, but this time instead of keeping the device stationary we need to *rotate* the device by a fixed amount of angle (θ). In our setting, we set $\theta = 180^\circ$ (or π `rad`). For example, Figure 9(b) shows how we rotate the smartphone by 180° around the positive x-axis.

[10]However, we found that a gyroscope's offset was impacted by orientation.

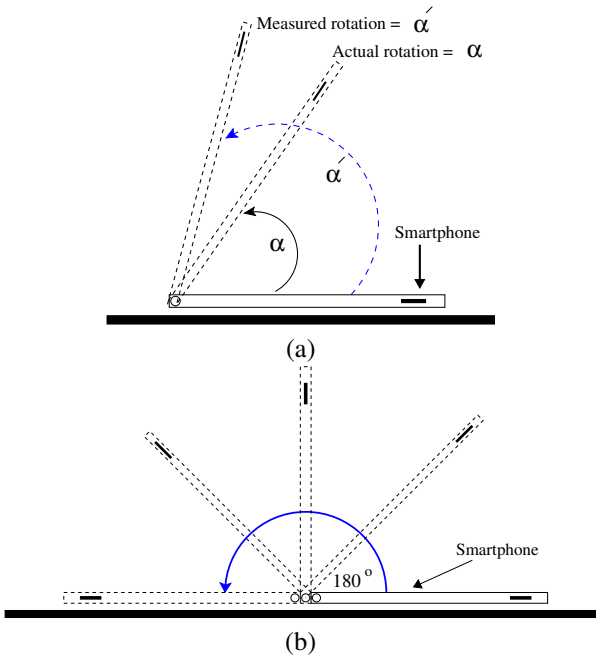

Fig. 9: a) Offset and gain error in gyroscope impact systems that use them for angular-displacement measurements. b) Calibrating the gyroscope by rotating the device 180° in the positive x-axis direction.

The angular displacement along any direction can be computed from gyroscopic data in the following manner:

$$\begin{aligned} \omega_i^M &= O_i + S_i\omega, \quad i \in \{\pm x, \pm y, \pm z\} \\ \int_0^t \omega_i^M \, \mathrm{d}t &= \int_0^t O_i \, \mathrm{d}t + S_i \int_0^t \omega \, \mathrm{d}t \\ \theta_i^M &= O_i t + S_i \theta \end{aligned} \quad (10)$$

where t refers to the time it took to rotate the device by θ angle with a fixed angular velocity of ω. Now, for any two measurements along the opposite directions of an axis we can compute the offset and gain error using the following equation:

$$O_i = \frac{\theta_{i+}^M + \theta_{i-}^M}{t_1 + t_2}, \quad S_i = \frac{\theta_{i+}^M - \theta_{i-}^M - O_i(t_1 - t_2)}{2\pi} \quad (11)$$

where $i \in \{x, y, z\}$ and t_1 and t_2 represents the timespan of the positive and negative measurement respectively. We take 10 measurements along all six directions $(\pm x, \pm y, \pm z)$ and compute the average offset and gain error along all three axes. However, since it is practically impossible to manually

rotate the device at a fixed angular velocity, the integration in equation (10) will introduce noise and therefore, the calculated errors will at best be approximations of the real errors. We also approximate the integral using *trapezoidal rule* which will introduce more error.

We next visualize the offset and gain error obtained from the gyroscopes of 30 smartphones (only showing for $\texttt{z}-\texttt{axis}$ where each color code represents a certain make-and-model). Figure 10 shows our findings. We see similar result compared to accelerometers where devices are scattered around at different regions of the plot. This suggests that gyroscopes exhibit different range of offset and gain error across different units.

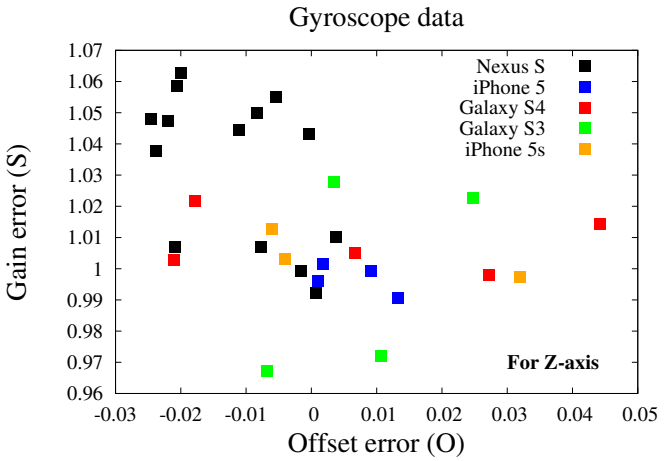

Fig. 10: Gyroscope offset and gain error from 30 smartphones.

Fingerprinting Calibrated Data: In this section we look at how calibrating sensors impact fingerprinting accuracy. For this setting, we first correct the raw values by removing the offset and gain errors before extracting features from them. That is, the calibrated value $a^C = (a^M - o)/g$. We then generate fingerprints on the *corrected data* and train the classifiers on the new fingerprints. Table XI shows the average F-score for calibrated data under three scenarios, considering both cases where the devices were kept on top of a desk and in the hand of a user. When we compare the results from uncalibrated data (Table V) to those from calibrated data, we see that the F-score reduces by approximately 16–25% for accelerometer data but not as much for the gyroscope data. This suggests that we were able to calibrate the accelerometer much more precisely than the gyroscope, as expected given the more complex and error-prone manual calibration procedure for the gyroscope. Another interesting observation is that audio stimulation provides small improvement in classifier accuracy. This suggests that audio stimulation does not influence the dominant features removed by the calibration, but does significantly impact secondary features that come into play once calibration is carried out. Overall, our results demonstrate that calibration is a promising technique, especially if more precise measurements can be made. Manufacturers should be encouraged to perform better calibration to both improve the accuracy of their sensors and to help protect users' privacy.

TABLE XI: Average F-score for calibrated data under lab setting

Device Placed	Stimulation	Avg. F-score (%)		
		Accelerometer	Gyroscope	Accelerometer+Gyroscope
On Desk	No-audio	71	97	97
	Sine	75	98	98
	Song	77	99	99
In Hand	No-audio	69	85	91
	Sine	70	90	93
	Song	69	89	93

B. Data Obfuscation

Rather than removing calibration errors, we can instead add extra noise to hide the miscalibration. This approach has the advantage of not requiring a calibration step, which requires user intervention and is particularly difficult for the gyroscope sensors. As such, the obfuscation technique could be deployed with an operating system update. Obfuscation, however, adds extra noise and can therefore negatively impact the utility of the sensors (in contrast to calibration, which improves their utility). In this section we will discuss the following techniques for adding noise –

- Uniform noise: highest entropy while having a bound.
- Laplace noise: highest entropy which is inspired by Differential Privacy.
- White noise: affecting all aspects of a signal.

1) Uniform Noise: In this section we randomly choose offset and gain errors from a uniform range where we deduce the base range from our lab phones.

Basic Obfuscation: First, we consider small obfuscation values in the range that is similar to what we observed in the calibration errors above. Adding noise in this range is roughly equivalent to switching to a differently (mis)calibrated phone and therefore should cause minimal impact to the user. To add obfuscation noise, we compute $a^O = (a^M - o^O)/g^O$, where g^O and o^O are the obfuscation gain and offset, respectively. Based on Figures 8 and 10, we choose a range of [-0.5,0.5] for the accelerometer offset, [-0.1,0.1] for the gyroscope offset, and [0.95,1.05] for the gain. For each session, we pick uniformly random obfuscation gain and offset values from the range; by varying the obfuscation values we make it difficult to fingerprint repeated visits. Table XII summarizes our findings when we apply obfuscation to all the sensor data obtained from our 30 lab smartphones. Compared to unaltered data (Table V), data obfuscation seems to provide significant improvement in terms of reducing the average F-score. Depending on the type of audio stimulation, F-score reduces by almost 7–24% when smartphones are kept stationary on the desk and by 23–42% when smartphones are kept stationary in the hand of the user. The impact of audio stimulation in fingerprinting motion sensors is much more visible in these results. We see that F-score increases by almost 18–21% when a song is being played in the background (compared to the no-audio scenario); again, we expect this to be a consequence of audio-stimulation significantly impacting secondary features that come into play once primary features are obfuscated.

Next, we apply similar techniques to the public and combined dataset. We apply the same range of offset and gain errors to the raw values before generating fingerprints. Table XIII and Table XIV summarizes our results for both

TABLE XII: Average F-score for obfuscated data under lab setting

Device Placed	Stimulation	Avg. F-score (%)		
		Accelerometer	Gyroscope	Accelerometer+Gyroscope
On Desk	No-audio	43	73	75
	Sine	49	76	76
	Song	71	88	93
In Hand	No-audio	46	46	51
	Sine	42	49	57
	Song	55	63	72

presence and absence of audio stimulation. We see that F-score reduces by approximately 20–41% (compared to Table VI and Table VII). We expect one of the reasons for the lower accuracy is the usage of a larger dataset, suggesting that for even larger sets the impact of obfuscation is likely to be even more pronounced.

TABLE XIII: Average F-score for obfuscated data under public setting where smartphones were kept *on top of a desk*

Stimulation	Avg. F-score (%)		
	Accelerometer	Gyroscope	Accelerometer+Gyroscope
No-audio	27	52	57
Sine	40	65	66

TABLE XIV: Average F-score for obfuscated data under both lab and public setting where smartphones were kept *on top of a desk*

Stimulation	Avg. F-score (%)		
	Accelerometer	Gyroscope	Accelerometer+Gyroscope
No-audio	26	50	55
Sine	41	69	75

Increasing Obfuscation Range: We now look at how the fingerprinting technique reacts to different ranges of obfuscation. Starting with our base ranges of $[-0.5, 0.5]$ and $[-0.1, 0.1]$ for the accelerometer and gyroscope offsets, respectively, and $[0.95, 1.05]$ for the gain, we linearly scale the ranges and observe the impact on F-score. We scale all ranges by the same amount, increasing the ranges symmetrically on both sides of the interval midpoint.

For this experimental setup we only consider the combined dataset as this contains the most number of devices (93 in total). We also restrict ourselves to the setting where we combine both the accelerometer and gyroscope features because this provides the best result (as evident from all our past results). Figure 11 highlights our findings. As we can see increasing the obfuscation range does reduce F-score but it has a *diminishing return*. For 10x increment, the F-score drops down to approximately 40% and 55% for no-audio and audio stimulation respectively. Beyond 10x increment (not shown) the reduction in F-score is minimal (at most 10% reduction at 50x increment). This result suggests that simply obfuscating the raw values is not sufficient to hide all unique characteristics of the sensors. So far we have only manipulated the signal value but did not alter any of the frequency features and as a result the classifier is still able to utilize the spectral features to uniquely distinguish individual devices.

Enhanced Obfuscation: Given that we know that the spectral features are not impacted by our obfuscation techniques, we now focus on adding noise to the frequency of the sensor signal. Our data injection procedure is described in Algorithm 1.

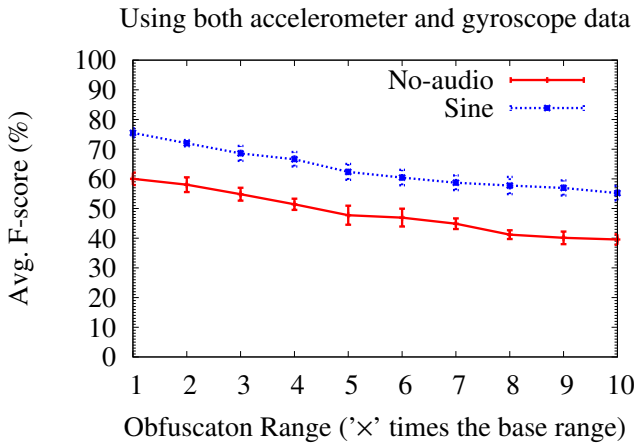

Fig. 11: Impact of obfuscation range as the range is linearly scaled up from 1x to 10x of the base range.

The main idea is to probabilistically insert a modified version of the current data point in between the past and current timestamp where the timestamp itself is randomly selected. Doing so will influence cubic interpolation of the data stream which in turn will impact the spectral features extracted from the data stream.

Algorithm 1 Obfuscated Data Injection

Input: Time series Data (D, T), Probability Pr, Offset O, Gain G, Offset Range O_{range}, Gain Range G_{range}
Output: Modified time series Data (MD, MT)
$offset \leftarrow Null$
$gain \leftarrow Null$
\# $Random(range)$: randomly selects a value in range
$j \leftarrow 1$
for $i = 1$ to $length(D)$ **do**
  #New data insertion
  if $i > 1$ and $Random([0,1]) < Pr$ **then**
    $offset \leftarrow Random(O_{range})$
    $gain \leftarrow Random(G_{range})$
    $MT[j] \leftarrow Random([T[i], MT[j-1])$
    $MD[j] \leftarrow (D[i] - offset)/gain$
    $j \leftarrow j + 1$
  end if
  #Original Data
  $MD[j] \leftarrow (D[i] - O)/G$
  $MT[j] \leftarrow T[i]$
  $j \leftarrow j + 1$
end for
return (MD, MT)

To evaluate our approach we first fix an obfuscation range. We choose 10x of the base range from the previous section as our fixed obfuscation range. We then vary the probability of data injection from [0,1]. Figure 12 shows our findings. We can see that even with relatively small amount of data injection (in the order of 20–40%) we can reduce the average F-score to approximately 15–20% depending on the type of input stimulation applied.

Impact of Uniform Noise on Utility: In this section we

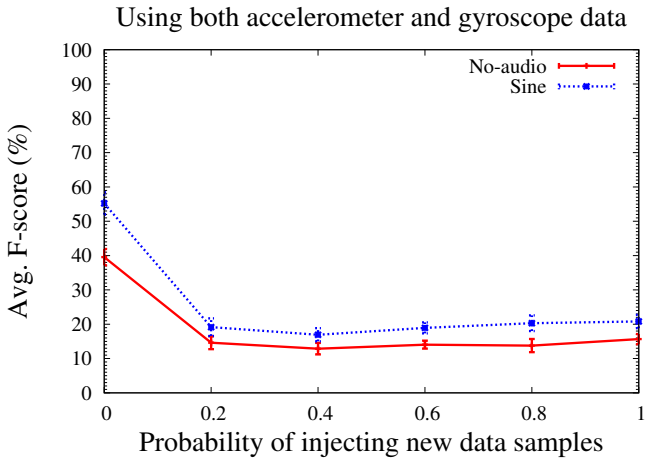

Fig. 12: Impact of randomly inserting new data points.

briefly analyze how uniform noise impact applications using motion sensors. To evaluate this we prototype a *Step Counter* application, a very popular smartphone application [60], that uses accelerometer readings to determine the number of steps taken by a user. We use the same procedure to collect sensor data through a web page. In our experimental setting, we ask the participant to take 20 steps while holding the phone in his/her hand and this whole process in repeated 10 times. We then calibrate[11] and obfuscate[12] the collected sensor data. Table XV shows the step counts computed from the original and modified sensor streams. Neither calibration nor basic obfuscation have a significant effect on accuracy. We would expect calibration to generally *improve* accuracy, but our calibration process is imperfect and it is possible that it introduces very minor errors. Basic obfuscation introduces errors that are commensurate with calibration errors of actual devices and thus also has minimal impact on accuracy. Increasing the obfuscation range introduces errors that are still within acceptable range. However, introducing new data points makes the accelerometer readings significantly less reliable, and we observe this effect in the step count. We next explore several alternative ways to add noise and their impact on privacy and utility.

TABLE XV: Impact of calibration and obfuscation

Stream Type	Step Count	
	Mean	Std dev
Original Stream	20	0
Calibrated Stream	20.1	0.32
Basic Obfuscation	20.1	0.32
Increased-Range Obfuscation	19.9	1.69
Enhanced Obfuscation	25.1	4.63

2) Laplace Noise: Next, we adopted an approach similar to differential privacy where we randomly selected offset and gain error from a Laplace distribution. From the definition of differential privacy [61], we know that a randomized function K gives ϵ-differential privacy if for all data sets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(K)$,

$$\Pr[K(D_1) \in S] \leq e^{\epsilon} \Pr[K(D_2) \in S] \quad (12)$$

We can remap this setting into our own problem where we can think of each device as a single data set, and K as the process of selecting random offset and gain error. S then becomes the outcome of applying random noise to raw sensor data. By changing ϵ we can control to what extent two device-output distributions are alike. In our setting we have offset and gain errors along 6 axes (xyz-axes for both accelerometer and gyroscope), giving us a total of 12 dimensions. We equally distribute our privacy budget ϵ along all 12 dimensions and select noise along the $i-th$ dimension using the following Laplace distribution: $Lap(0, \beta_i)$ where $\beta_i = S_i/(\epsilon/12)$ and $S_i = \max(\text{i-th Dimensional values}) - \min(\text{i-th Dimensional values})$, $i \in \{1, 2, ..., 12\}$. Figure 13 shows that as we increase ϵ (i.e., as we lower the scale parameter of the Laplace distribution), F-score also increases. But even with a relatively high privacy budget of $\epsilon = 10$ we see that F-score reduces from around 95% to 47–65% depending on the type of background stimulation.

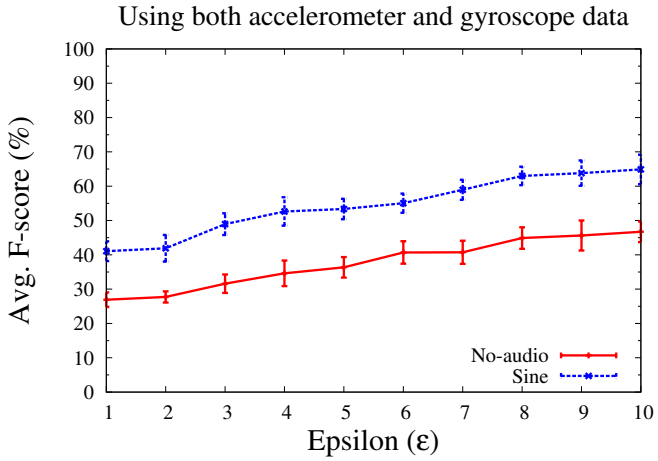

Fig. 13: Randomly selecting offset/gain errors from a Laplace distribution.

Impact of Laplace Noise on Utility: We rerun our step counter application on sensor data where we select offset and gain error from a Laplace distribution while varying ϵ. Figure 14 shows how step count evolves for different levels of privacy budget (ϵ). We see that as we increase ϵ, step count converges to the expected value with negligible deviation. For $\epsilon \geq 6$ the confidence interval is negligible, i.e., for $\epsilon \geq 6$ the impact of noise is minimal. Notably, on Figure 13, we can see that for $\epsilon = 6$, we get significantly lower classification accuracy than using low levels of uniform noise (see Figure 11). This suggests that Laplace noise may achieve a better tradeoff between privacy and utility; we plan to investigate its impact on the utility of other applications in the future.

3) White Noise: From figure 13 we see that even when $\epsilon = 1$ we can achieve an F-score of 26–41%. We then looked at the dominant features and found that spectral features like *spectral irregularity*, *spectral attack slope* and *spectral entropy* are dominant. Changing the gain and offset have minimal

[11]Using a handset for which we have computed calibration errors.
[12]Using random offset and gain error for each session.

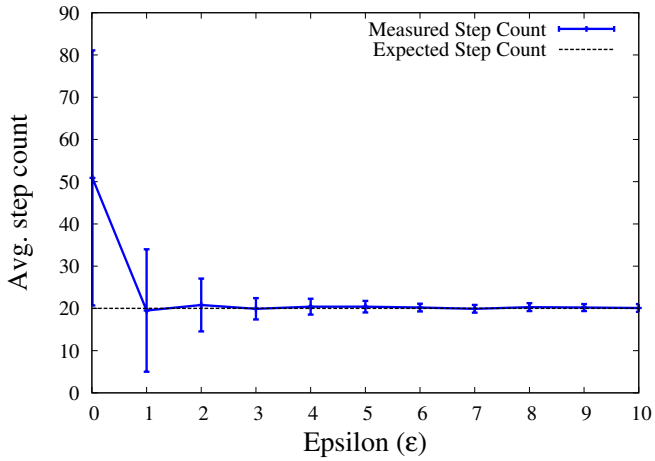

Fig. 14: Impact of Laplace noise on utility.

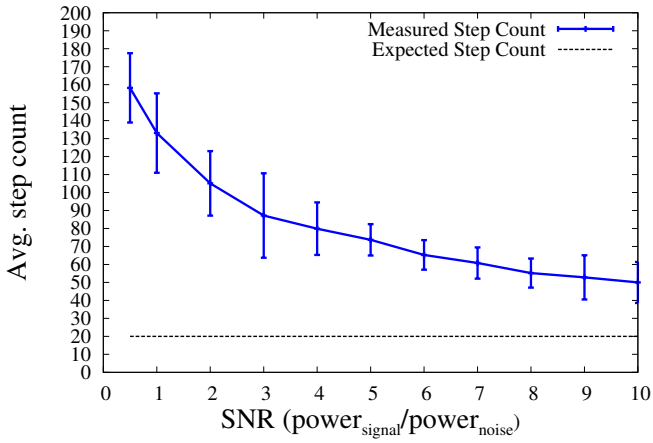

Fig. 16: Impact of white noise on sensor utility.

impact on spectral features; therefore we next added Gaussian white noise to the signal, after applying random offset and gain error from a Laplace distribution. For this experimental setup we fixed $\epsilon = 6$ (because for $\epsilon = 6$ we observed minimal impact on utility in Figure 14) and varied the signal-to-noise ratio (SNR). Figure 15 highlights F-score for different values of SNR. We can see that F-score remains more or less steady but increases slightly for higher SNRs. However, compared to Laplace noise (Figure 13) we see that F-score decreases significantly when white noise is added to the signal.

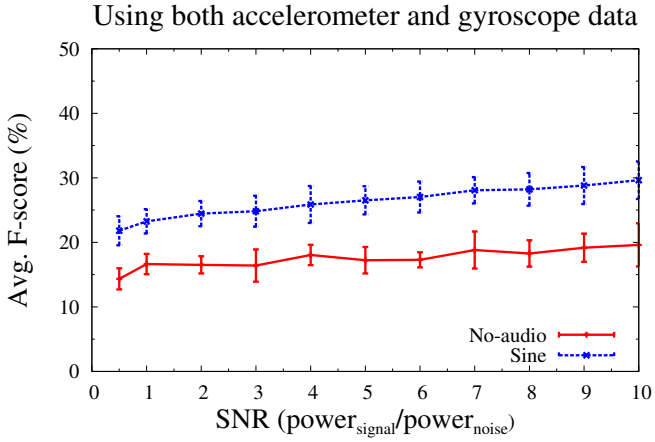

Fig. 15: Impact of white noise on F-score.

Impact of White Noise on Utility: Given that we see adding white noise provides low F-scores we wanted to see what kind on impact it would have on sensor utility. To evaluate this we rerun our step counter application on sensor data after applying Gaussian white noise. Figure 16 highlights the computed step counts for different SNRs. We see that adding white noise has drastic consequences as it increases the number of steps counted significantly, even at high signal-to-noise ratios.

C. Deployment Considerations

We envision our obfuscation technique as an update to the mobile operating system. Under default setting, data is always obfuscated unless the user explicitly allows an application to access unaltered sensor data. As we just observed for some applications small amount of obfuscation does not impact their utility, however, for others, e.g., a 3-D game might need access to raw accelerometer and gyroscope data instead of the obfuscated data to operate properly, in which case this will be noticeable to the user who can then provide the appropriate permission to the application. Our default obfuscated-setting will ensure that users do not have to worry about applications like browser accessing sensor data without their awareness.

VII. Limitations

Our approach has a few limitations. First, we experimented with 93 devices; a larger target device pool could lower our accuracy. However, we conducted our experiments in real-world settings (i.e., users under natural web browsing settings), collecting data from a wide variety of smartphones. We, therefore, believe our results are representatives of real-world scenarios. Secondly, our calibration process has some errors, specially the manual calibration process for the gyroscope is error-prone as it is impossible to manually rotate the device at a fixed angular velocity. That being said one of our main goals is to show that even simple calibration techniques can reasonably reduce device fingerprinting.

VIII. Conclusion

In this paper, we show that motion sensors such as accelerometers and gyroscopes can be used to uniquely identify smartphones. The more concerning matter is that these sensors can be surreptitiously accessed by a web page publisher without users' awareness. We also show that injecting audio stimulation in the background improves detection rate as sensors like gyroscopes react to acoustic stimulation uniquely.

Our countermeasure techniques, however, mitigate such threats by obfuscating anomalies in sensor data. We were able to significantly reduce fingerprinting accuracy by employing simple, yet effective obfuscation techniques. As a general conclusion, we suggest using our obfuscation techniques in the absence of explicit user permission/awareness.

ACKNOWLEDGMENT

We would like to thank all the anonymous reviewers for their valuable feedback. We would specially like to thank Romit Roy Choudhury and his group at UIUC for providing us with the bulk of the smartphones used in our experiments. On the same note we would like to extend our gratitude to the Computer Science department at UIUC for providing us with the remaining smartphones used in our experiments. We give special thanks to all the participants who took the time to participate in our online data collection study. This paper reports on work that was supported in part by NSF CNS 1053781 and NSF CNS 0953655.

REFERENCES

[1] K. Mowery and H. Shacham, "Pixel perfect: Fingerprinting canvas in HTML5," in *Proceedings of Web 2.0 Security and Privacy Workshop (W2SP)*, 2012.

[2] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable," in *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS)*, 2014.

[3] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile Device Identification via Sensor Fingerprinting," *CoRR*, vol. abs/1408.1416, 2014. [Online]. Available: http://arxiv.org/abs/1408.1416

[4] A. Ross and A. Jain, "Information fusion in biometrics," *Pattern Recognition Letters*, vol. 24, no. 13, pp. 2115 – 2125, 2003.

[5] S. COLE and S. Cole, *Suspect Identities: A History of Fingerprinting and Criminal Identification*. Harvard University Press, 2009.

[6] L. Langley, "Specific emitter identification (SEI) and classical parameter fusion technology," in *Proceedings of the IEEE WESCON*, 1993, pp. 377–381.

[7] M. Riezenman, "Cellular security: better, but foes still lurk," *IEEE Spectrum*, vol. 37, no. 6, pp. 39–42, 2000.

[8] Z. Li, W. Xu, R. Miller, and W. Trappe, "Securing Wireless Systems via Lower Layer Enforcements," in *Proceedings of the 5th ACM Workshop on Wireless Security (WiSe)*, 2006, pp. 33–42.

[9] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng, "Device fingerprinting to enhance wireless security using nonparametric Bayesian method," in *Proceedings of the 30th Annual IEEE International Conference on Computer Communications (INFOCOM)*, 2011, pp. 1404–1412.

[10] N. Patwari and S. K. Kasera, "Robust Location Distinction Using Temporal Link Signatures," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2007, pp. 111–122.

[11] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless Device Identification with Radiometric Signatures," in *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2008, pp. 116–127.

[12] R. M. Gerdes, T. E. Daniels, M. Mina, and S. F. Russell, "Device identification via analog signal fingerprinting: A matched filter approach," in *Proceedings of the 13th Network and Distributed System Security Symposium (NDSS)*, 2006.

[13] S. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *Proceedings of the 18th Annual IEEE International Conference on Computer Communications (INFOCOM)*, vol. 1, 1999, pp. 227–234.

[14] T. Kohno:2005, A. Broido, and K. C. Claffy, "Remote Physical Device Fingerprinting," *IEEE Trans. Dependable Secur. Comput.*, vol. 2, no. 2, pp. 93–108, 2005.

[15] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying Unique Devices Through Wireless Fingerprinting," in *Proceedings of the First ACM Conference on Wireless Network Security (WiSec)*, 2008, pp. 46–55.

[16] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting," in *Proceedings of the 15th Conference on USENIX Security Symposium*, 2006.

[17] F. Guo and T. cker Chiueh, "Sequence Number-Based MAC Address Spoof Detection," in *Proceedings of 8th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2005.

[18] G. Lyon. Nmap: a free network mapping and security scanning tool. http://nmap.org/.

[19] F. Yarochkin, M. Kydyraliev, and O. Arkin. Xprobe project. http://ofirarkin.wordpress.com/xprobe/.

[20] P. Eckersley, "How Unique is Your Web Browser?" in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies (PETS)*, 2010, pp. 1–18.

[21] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting Information in JavaScript Implementations," in *Proceedings of IEEE Web 2.0 Security & Privacy Workshop (W2SP)*, 2011.

[22] L. Olejnik, C. Castelluccia, and A. Janc, "Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns," in *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs)*, 2012.

[23] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel, "FPDetective: dusting the web for fingerprinters," in *Proceedings of the 2013 ACM SIGSAC conference on Computer and Communications Security (CCS)*, 2013, pp. 1129–1140.

[24] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web never forgets: Persistent tracking mechanisms in the wild," in *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 674–689.

[25] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "You are what you include: large-scale evaluation of remote javascript inclusions," in *Proceedings of the 19th ACM SIGSAC conference on Computer and Communications Security (CCS)*, 2012, pp. 736–747.

[26] N. Nikiforakis, W. Joosen, and B. Livshits, "PriVaricator: Deceiving Fingerprinters with Little White Lies," in *Proceedings of the 24th International Conference on World Wide Web (WWW)*, 2015, pp. 820–830.

[27] Apple places kill date on apps that use 'UDID' device identifiers. http://www.zdnet.com/article/apple-places-kill-date-on-apps-that-use-udid-device-identifiers/.

[28] Android TelephonyManager. http://developer.android.com/reference/android/telephony/TelephonyManager.html#getDeviceId().

[29] A. Das, N. Borisov, and M. Caesar, "Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components," in *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 441–452.

[30] Z. Zhou, W. Diao, X. Liu, and K. Zhang, "Acoustic Fingerprinting Revisited: Generate Stable Device ID Stealthily with Inaudible Sound," in *Proceedings of the 21st ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2014, pp. 429–440.

[31] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing Speech from Gyroscope Signals," in *Proceedings of the 23rd USENIX Conference on Security Symposium*, 2014, pp. 1053–1067.

[32] Y. Song, M. Kukreti, R. Rawat, and U. Hengartner, "Two Novel Defenses against Motion-Based Keystroke Inference Attacks," in *Workshop of Mobile Security Technologies (MoST) co-located with IEEE Symposium on Security and Privacy*, 2014.

[33] STMicroelectronics. http://www.st.com/web/en/home.html.

[34] Invensense. http://www.invensense.com/.

[35] Research and Markets: Global MEMS Market 2015-2019. http://www.businesswire.com/news/home/20150216005540/en/Research-Markets-Global-MEMS-Market-2015-2019--#.VOVr7HVGh5Q.

[36] iPhone 4 Teardown. https://www.ifixit.com/Teardown/iPhone+4+Teardown/3130.

[37] iPhone 5 Teardown. https://www.ifixit.com/Teardown/iPhone+5+Teardown/10525.

[38] iPhone 6 Teardown. https://www.ifixit.com/Teardown/iPhone+6+Teardown/29213.

[39] Inside the Samsung Galaxy SIII. http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/inside-the-samsung-galaxy-siii/.

[40] Inside the Samsung Galaxy S4. http://www.chipworks.com/en/technical-competitive-analysis/resources/blog/inside-the-samsung-galaxy-s4/.

[41] Nexus 4 Teardown. https://www.ifixit.com/Teardown/Nexus+4+Teardown/11781.

[42] Nexus 5 Teardown. https://www.ifixit.com/Teardown/Nexus+5+Teardown/19016.

[43] MEMS-based accelerometers. http://www.wikid.eu/index.php/MEMS-based_accelerometers.

[44] J. Seeger, M. Lim, and S. Nasiri. Development of High-Performance High-Volume consumer MEMS Gyroscope. http://www.invensense.com/mems/gyro/documents/whitepapers/Development-of-High-Performance-High-Volume-Consumer-MEMS-Gyroscopes.pdf.

[45] STMicroelectronics. Everything about STMicroelectronics 3-axis digital MEMS gyroscopes. http://www.st.com/web/en/resource/technical/document/technical_article/DM00034730.pdf.

[46] MEMS gyroscopes. http://www.findmems.com/wikimems-learn/introduction-to-mems-gyroscopes.

[47] S. McKinley and M. Levine, "Cubic Spline Interpolation," *College of the Redwoods*, vol. 45, no. 1, pp. 1049–1060, 1998.

[48] A. Das, N. Borisov, and M. Caesar, "Exploring Ways To Mitigate Sensor-Based Smartphone Fingerprinting," *CoRR*, vol. abs/1503.01874, 2015. [Online]. Available: http://arxiv.org/abs/1503.01874

[49] Supervised Learning (Machine Learning) Workflow and Algorithms. http://www.mathworks.com/help/stats/supervised-learning-machine-learning-workflow-and-algorithms.html.

[50] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.

[51] Percentage of all global web pages served to mobile phones. http://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share/.

[52] Top Mobile Browsers from Jan 2014 toJan 2015. http://gs.statcounter.com/#mobile_browser-ww-monthly-201401-201501.

[53] Browser Trends September 2014: Chrome Is the Top Mobile Browser. http://www.sitepoint.com/browser-trends-september-2014-chrome-top-mobile-browser/.

[54] Android Sensors Overview. http://developer.android.com/guide/topics/sensors/sensors_overview.html.

[55] Corona SDK API reference. http://docs.coronalabs.com/api/library/system/setAccelerometerInterval.html.

[56] MIRtoolbox. https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox.

[57] LibXtract Documentation. http://libxtract.sourceforge.net/.

[58] A. Pocock and G. Brown, "FEAST," 2014, http://mloss.org/software/view/386/.

[59] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján, "Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection," *Machine Learning Research*, vol. 13, pp. 27–66, 2012.

[60] Wearables vs. Smartphone Apps: Which Are Better to Count Steps? http://www.livescience.com/49756-smartphone-apps-wearables-step-counts.html.

[61] C. Dwork, "Differential Privacy," in *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*. Springer Verlag, 2006, pp. 1–12.

Appendix A
Accessing Motion Sensors From Browser

To access motion sensors the *DeviceMotion* class needs to be initialized. A sample JavaScript snippet is given below:

```
if(window.DeviceMotionEvent!=undefined){
    window.addEventListener('devicemotion',
       motionHandler);
    window.ondevicemotion = motionHandler;
}
```

```
function motionHandler(event){
    agx = event.accelerationIncludingGravity.x;
    agy = event.accelerationIncludingGravity.y;
    agz = event.accelerationIncludingGravity.z;
    ai = event.interval;
    rR = event.rotationRate;
    if (rR != null) {
            arAlpha = rR.alpha;
            arBeta = rR.beta ;
            arGamma = rR.gamma;
    }
}
```

Appendix B
Screenshot of Our Data Collection Webpage

We provide screenshots (see Figure 17) of our data collection website to give a better idea of how participants were asked to participate.

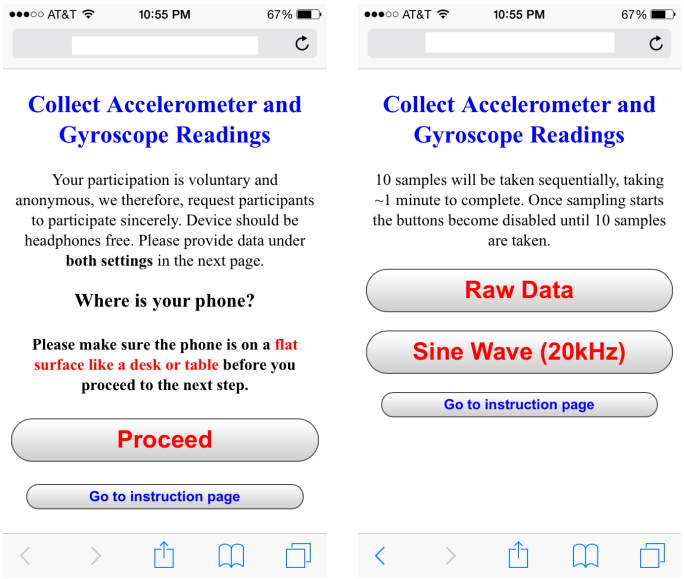

Fig. 17: Screenshot of our data collection website.